Scalable I/O Middleware and File System Optimizations for High-performance Computing

Wei-keng Liao, Alok Choudhary
Northwestern University

Mahmut Kandemir
Pen State University

Supported by NSF and DOE

HEC FSIO Workshop, 2008

Project Overview

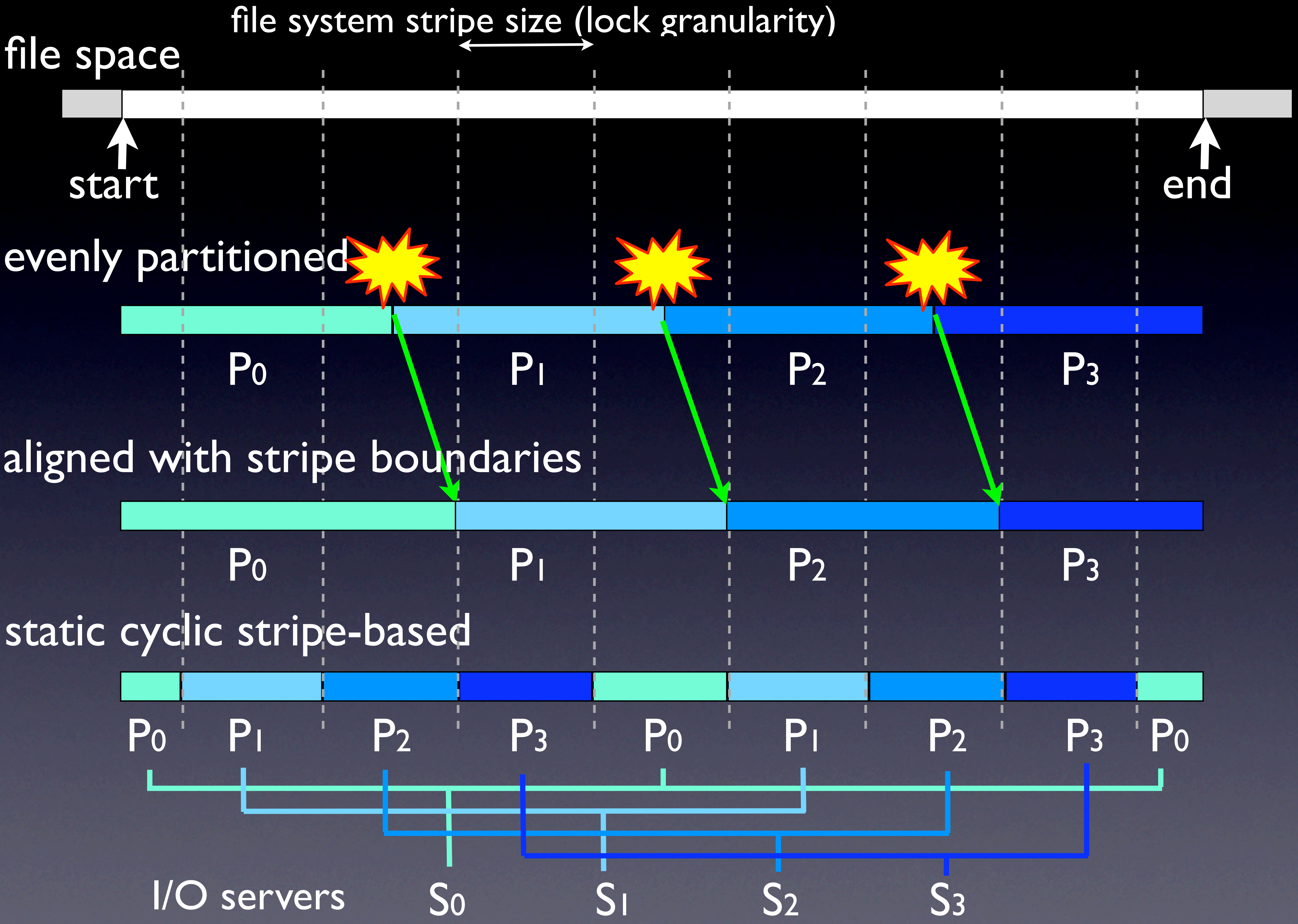
- Improving MPI I/O performance
  - Individual collective I/O operation
  - Across multiple I/O operations
- Improving caching/prefecting at I/O servers
  - Eliminate harmful prefetching and eviction

Unique vs. Shared file I/O

- Two programming styles for parallel apps
- Unique-file I/O usually performs better
  - ✦ No data consistency and cache coherence issues
  - ✦ Problem with file management
- Shared-file I/O produces less files
  - ✦ Easier for management, data are in canonical order
  - ✦ File systems must enforce data atomicity and coherent cache

MPI Collective I/O

- ROMIO uses the two-phase I/O strategy
  - ✦ Communication phase
    - ✳ Redistribute data among processes in a way the I/O phase is the least expensive
  - ✦ I/O phase
    - ✳ Fast when I/Os are large contiguous chunks of requests
- Can I/O phase perform like unique-file I/O?



file system stripe size (lock granularity)

file space

start

end

evenly partitioned

P_0 P_1 P_2 P_3

aligned with stripe boundaries

P_0 P_1 P_2 P_3

static cyclic stripe-based

P_0 P_1 P_2 P_3 P_0 P_1 P_2 P_3 P_0

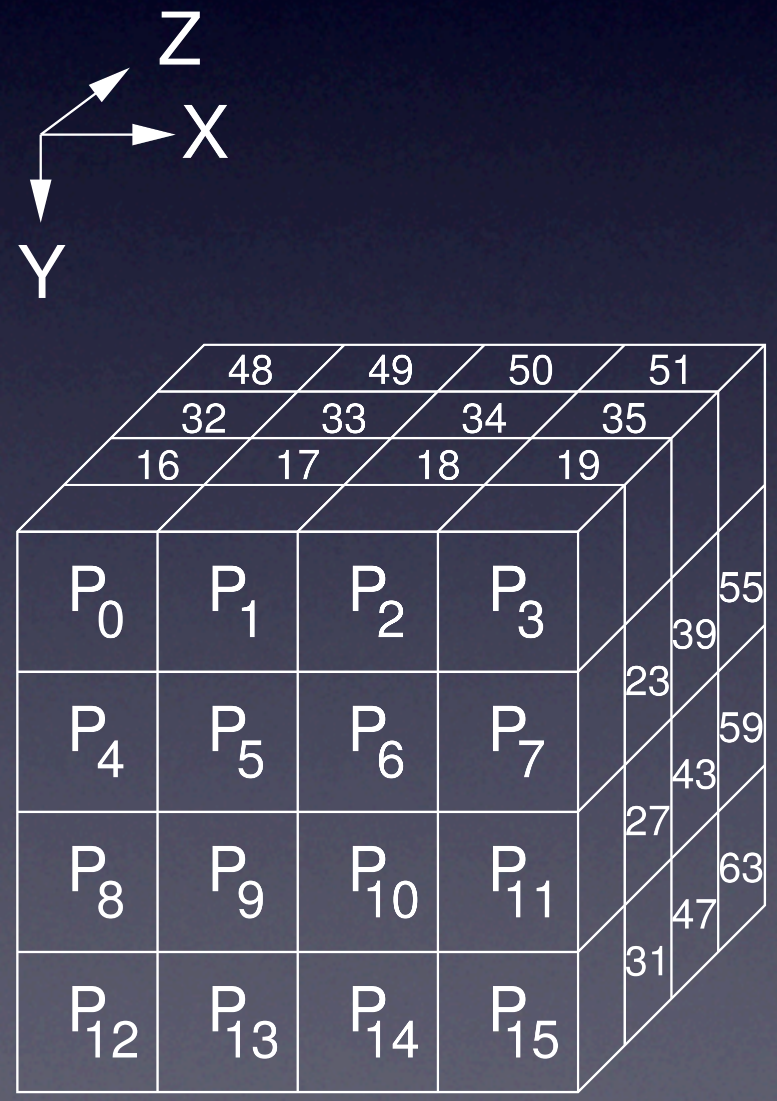
I/O servers S_0 S_1 S_2 S_3

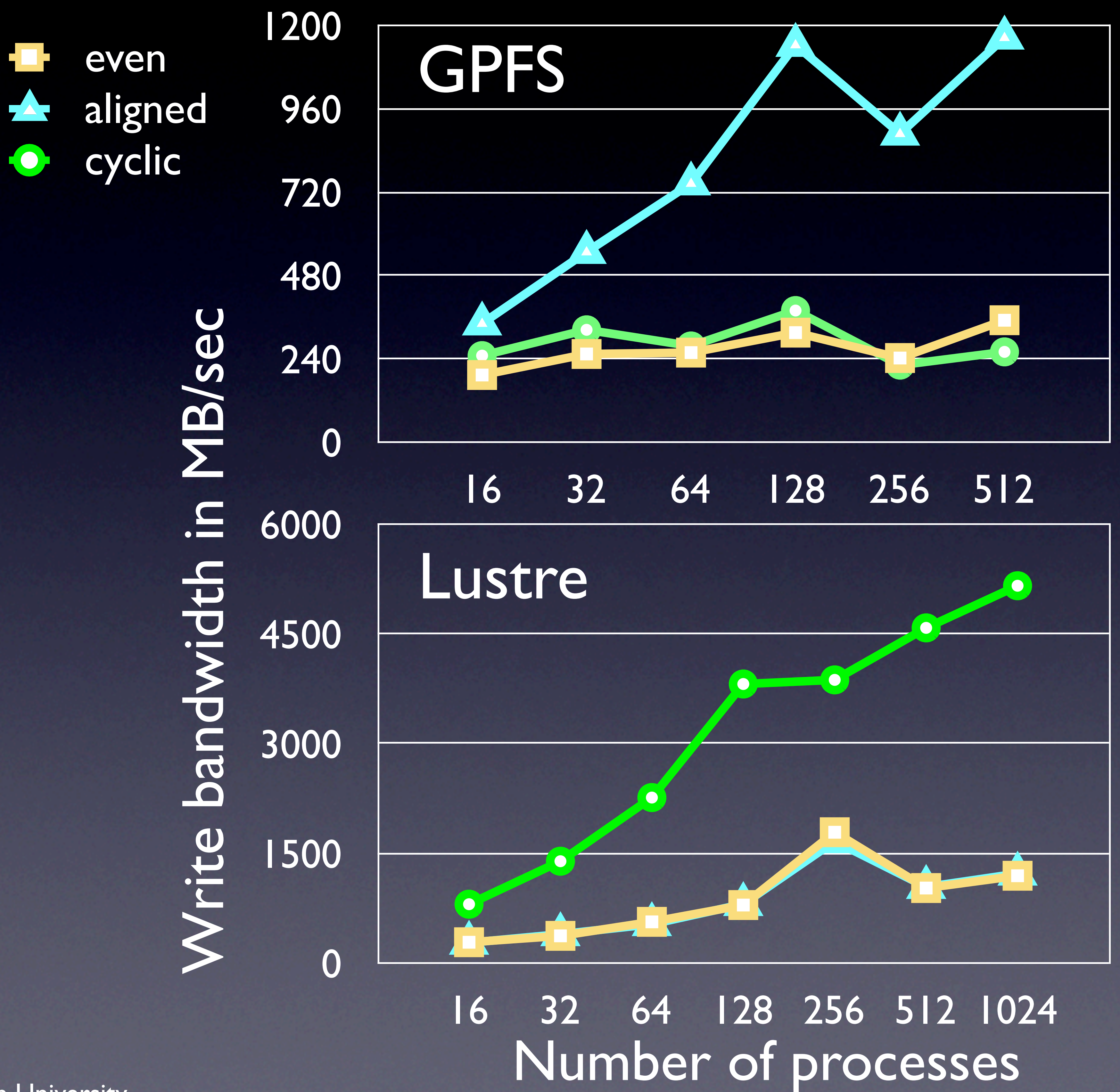
File Locking Protocols

- Token-based -- GPFS
  - A token holder has authority for granting further lock requests to its already-granted byte range
  - Mercury, IBM IA-64 Linux, TeraGrid, NCSA
  - Lock granularity == file stripe size
- Server-based -- Lustre
  - Each server manages locks for the file stripes it stores
  - Jaguar, Cray XT, ORNL
  - Lock granularity == file stripe size

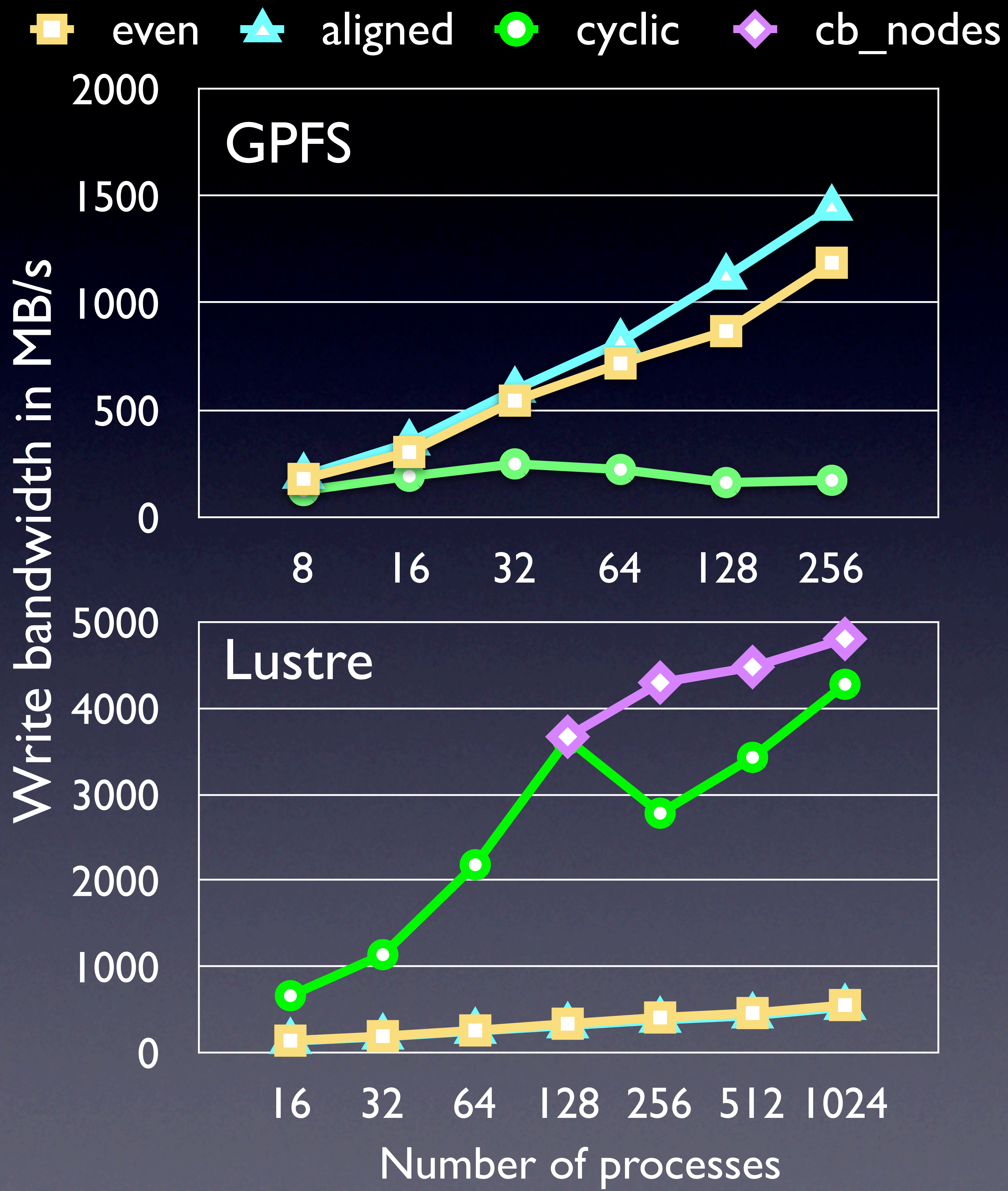
ROMIO test for collective I/O

3D block partitioning





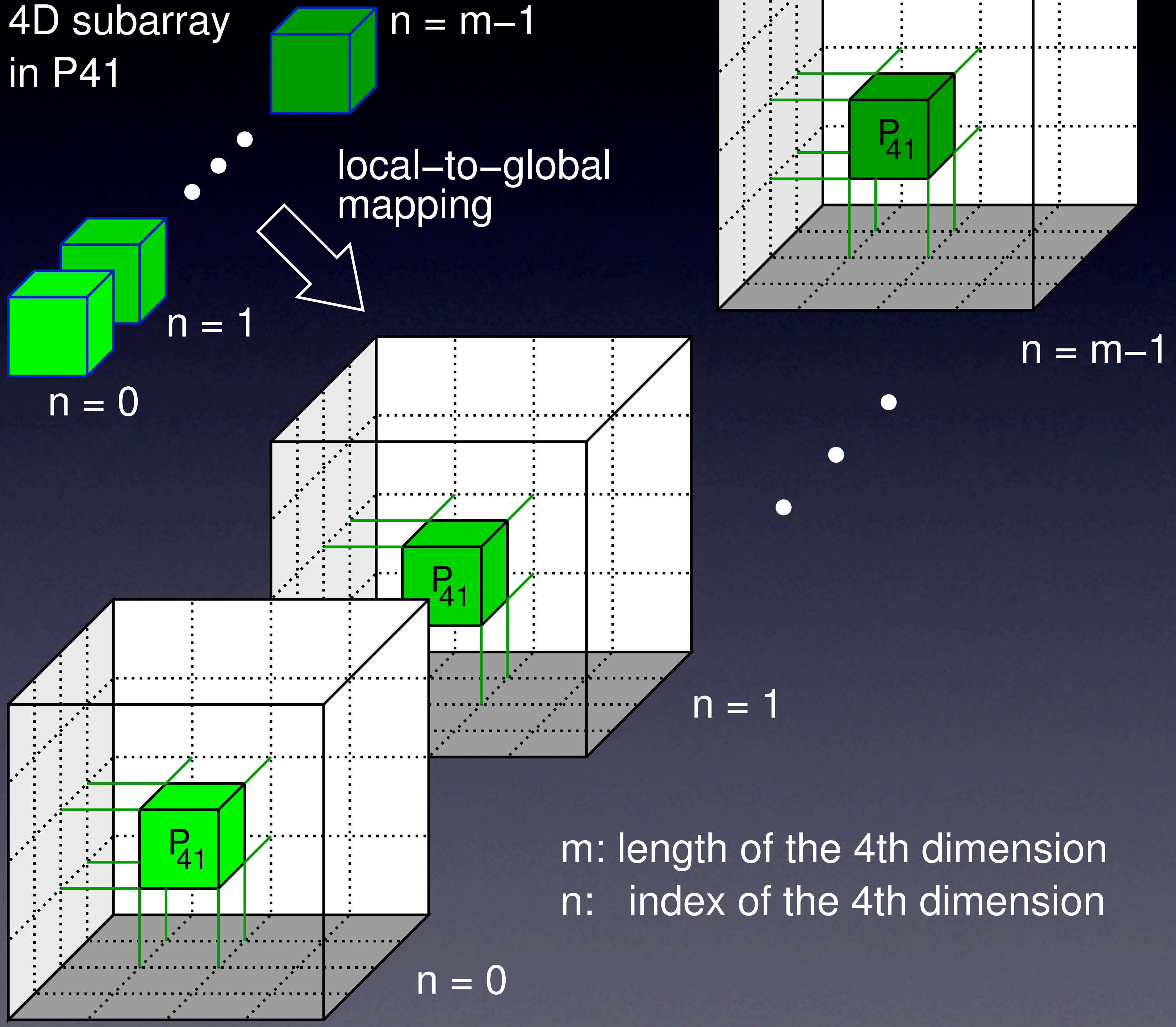
FLASH I/O



- I/O kernel of the FLASH application from University of Chicago
- I/O method: HDF5
  - Each process writes 80 32x32x32 arrays
  - I/O amount increases as the number of MPI processes
- I/O pattern
  - Non-interleaved among processes

S3D I/O Pattern

S3D is a turbulent combustion application using a direct numerical simulation solver from SNL

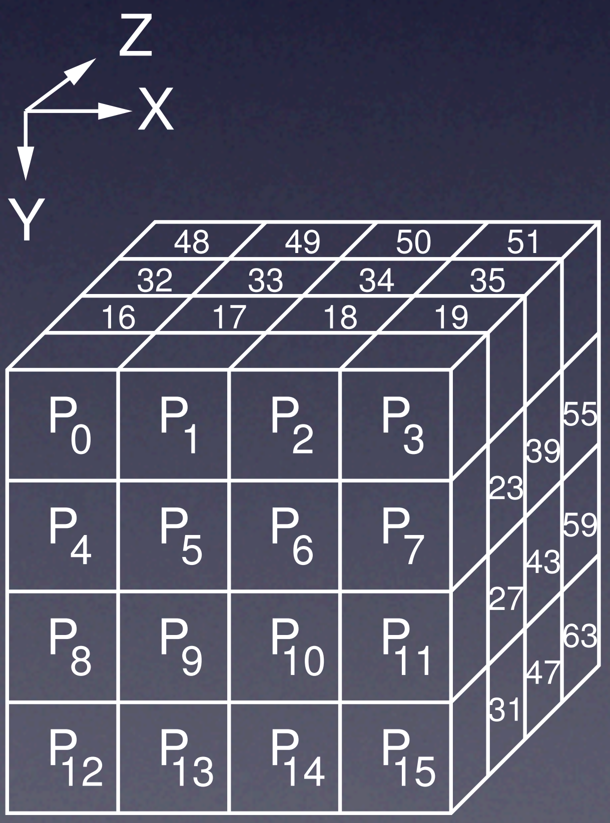


4D subarray in P41

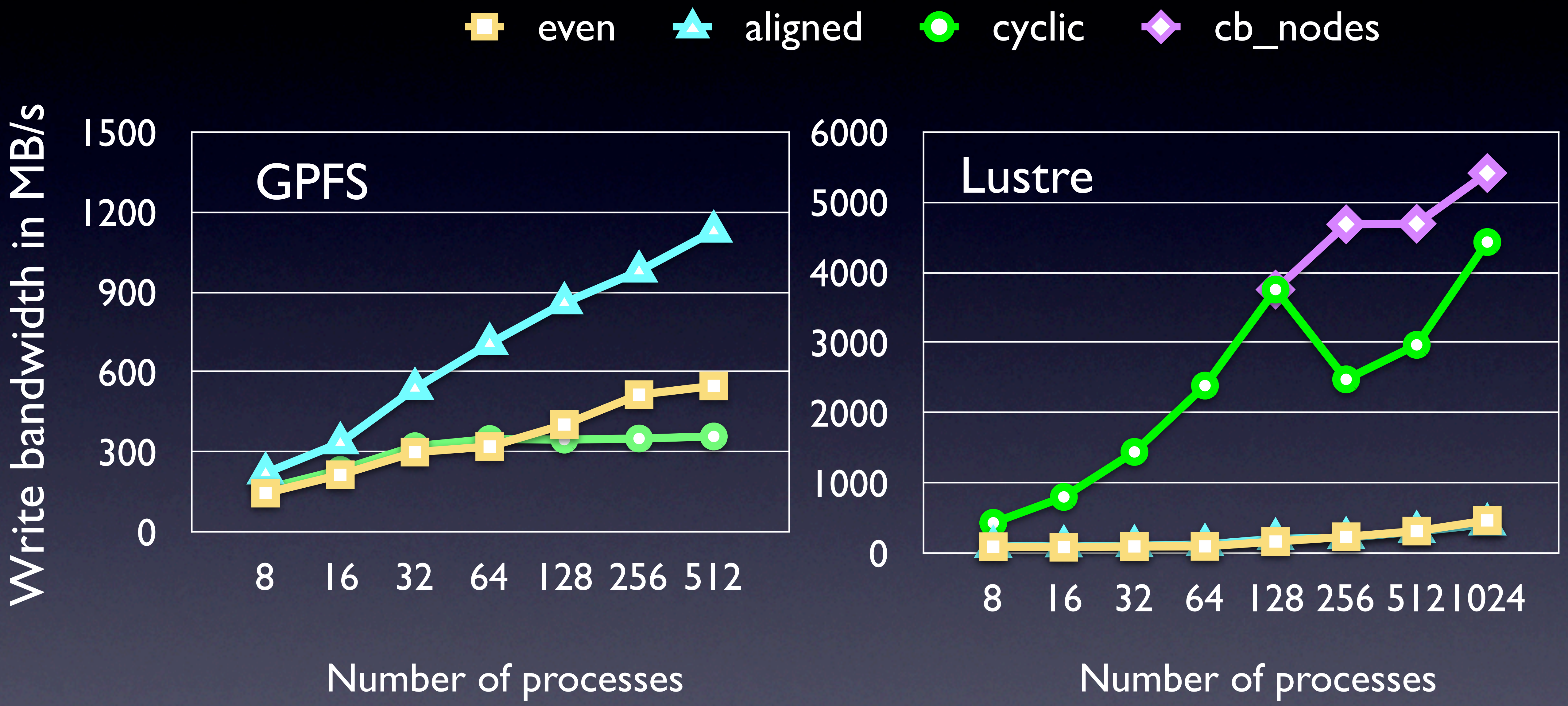
local−to−global mapping

m: length of the 4th dimension

n: index of the 4th dimension



S3D I/O


even
aligned
cyclic
cb_nodes
Write bandwidth in MB/s
GPFS
1500
1200
900
600
300
0
8
16
32
64
128
256
512
Number of processes
Lustre
6000
5000
4000
3000
2000
1000
0
8
16
32
64
128
256
512
1024
Number of processes

Summary I

- Token-based locking protocol -- GPFS
  - ✦ Use file domains that align with stripe boundaries
- Server-based locking protocol -- Lustre
  - ✦ Use static-cyclic partitioning method
  - ✦ Choose cb_nodes to be a multiple of stripe width
- Communication phase becomes important
  - ✦ Currently using MPI All-to-all and Isend/Irecv, they do not scale well beyond 1000 processes

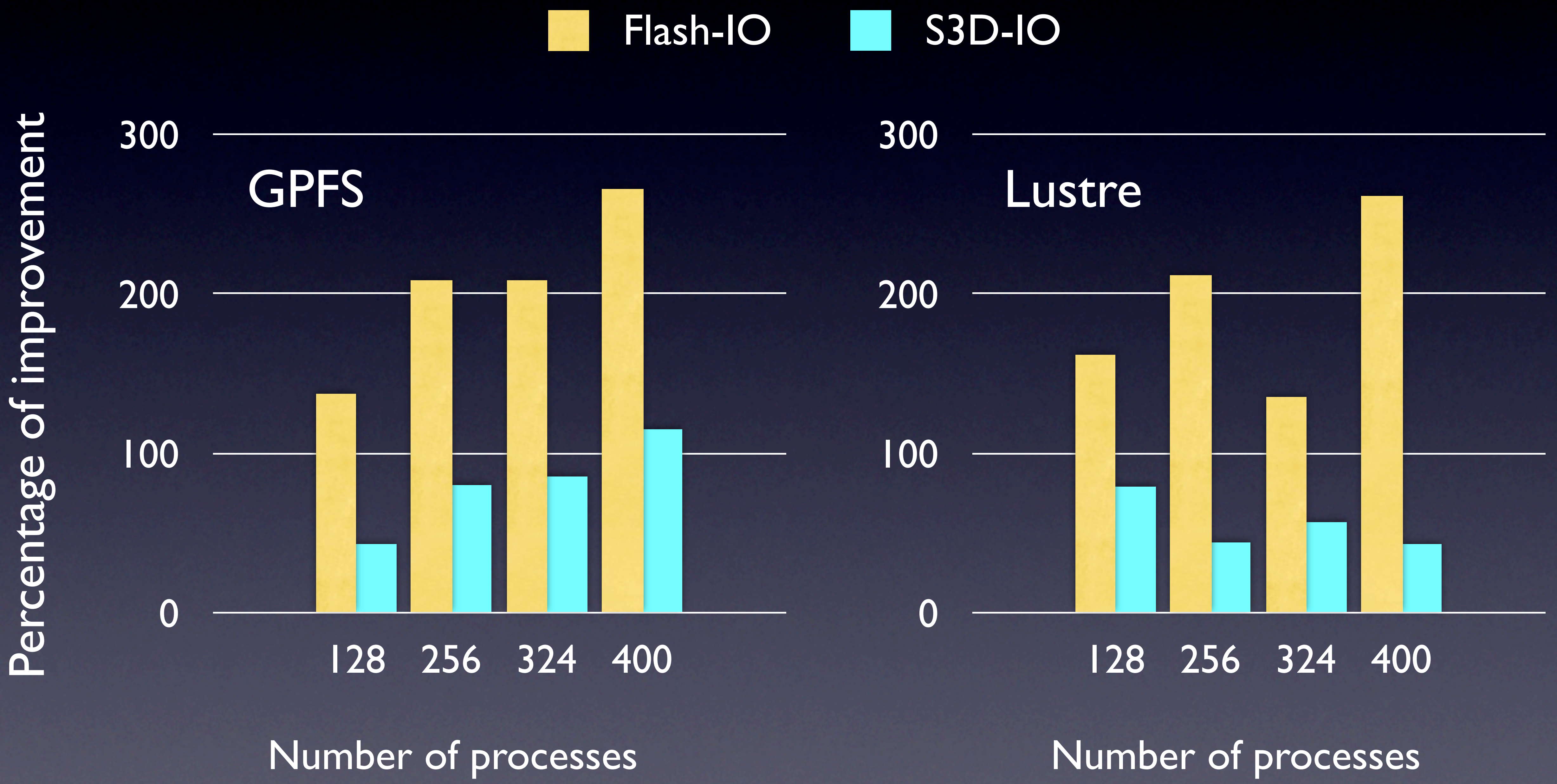
I/O Delegate

- Optimization considering multiple collective or independent I/O calls
- Allocate a separate group of compute nodes as I/O delegates
  - ✦ Uses a small percentage (< 10 %) of additional resource
  - ✦ Aggregate small requests to larger ones
  - ✦ Rearrange data based on file system locking protocols
  - ✦ Entire memory space can be used as collective buffer at delegates

Collaborated File Caching

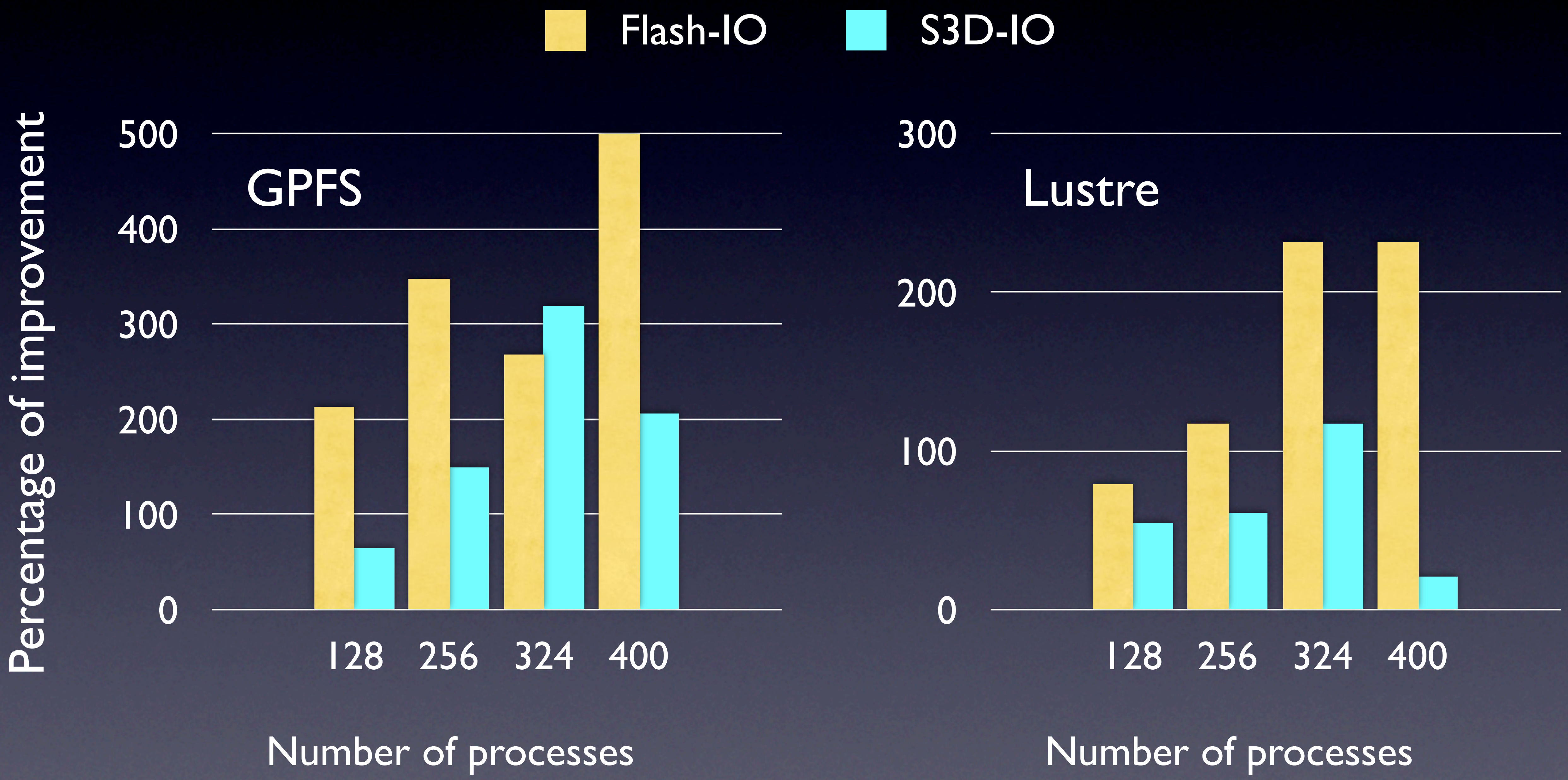
- A fully functional distributed, coherent cache system at the delegates
- Cache metadata management
  - ✦ Metadata are cyclically distributed among all processes
  - ✦ Lock protocol for metadata atomicity
- Caching policies
  - ✦ Local: page eviction (least-recent used)
  - ✦ Global: page migration (referred consecutively twice)

I/O Delegates are 3%



Wei-keng Liao, EECS Department, Northwestern University

I/O Delegates are 10%



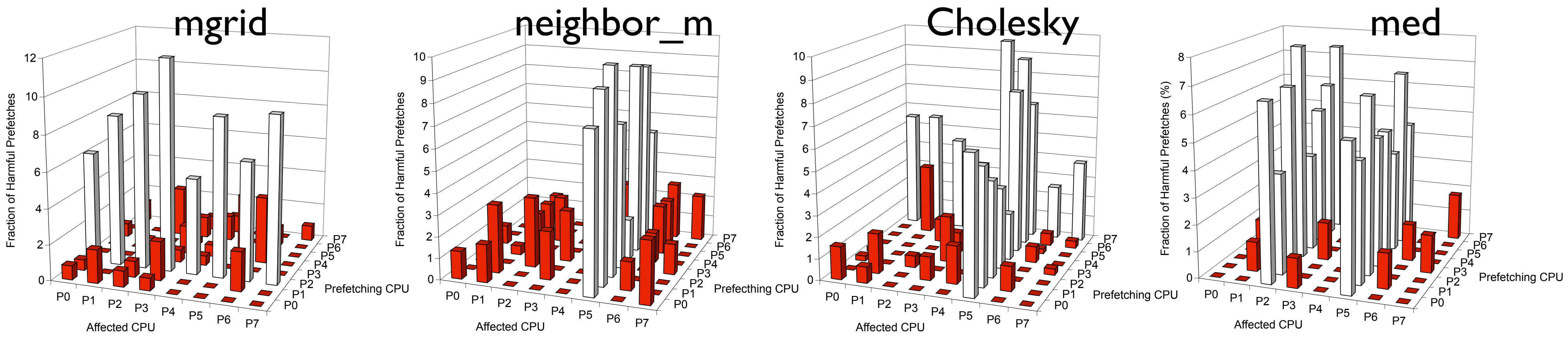
Summary II

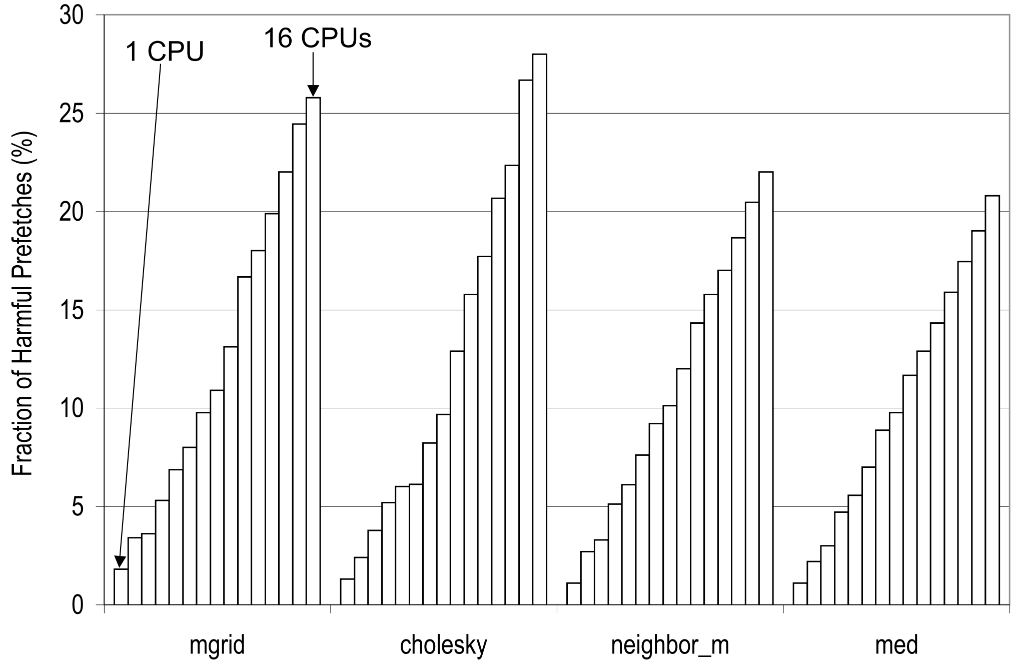
- I/O delegate is designed to improve multiple MPI I/O operations
    - ✦ Small percentage of additional nodes provides significant I/O improvement
- Future work
    - ✦ Integrate into two-phase I/O
    - ✦ Incorporate the file domain partitioning methods

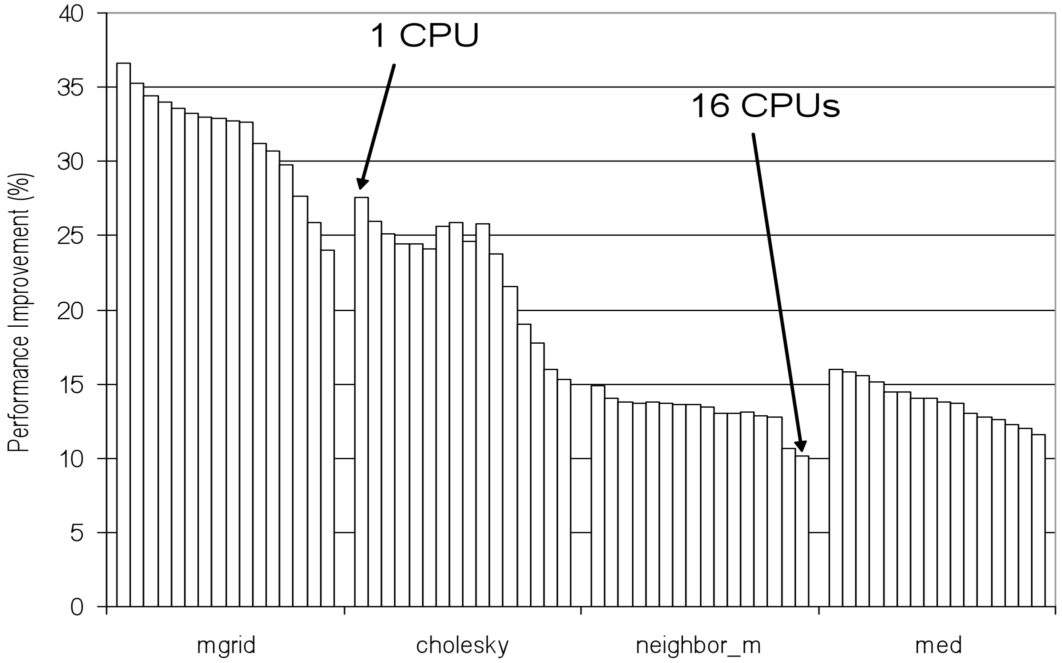
Data Throttling and Pinning

- In prefetch throttling, one or more CPUs are (temporarily) prevented from issuing prefetch requests to reduce the number of harmful prefetches

- In data pinning, select data blocks brought to the memory cache by a CPU are marked as non-removable (i.e., pinned in the cache) for a certain period of time

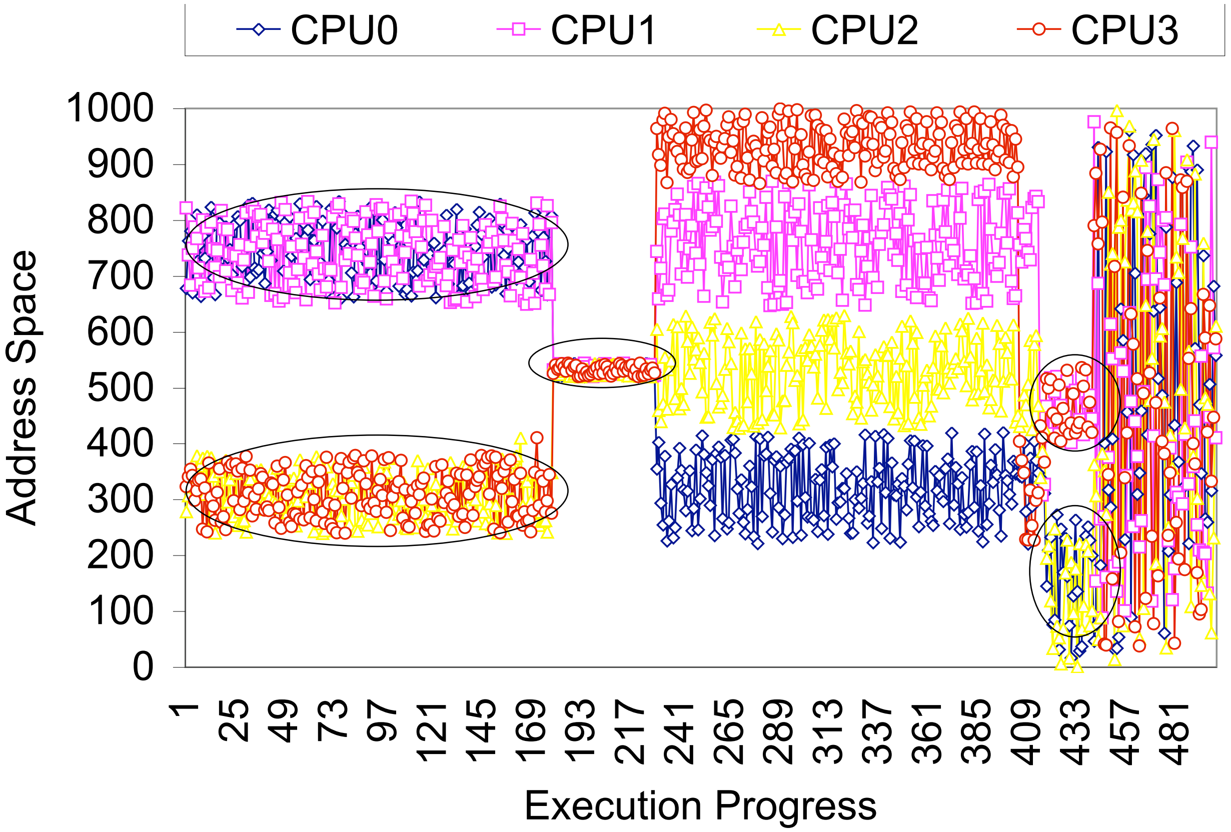


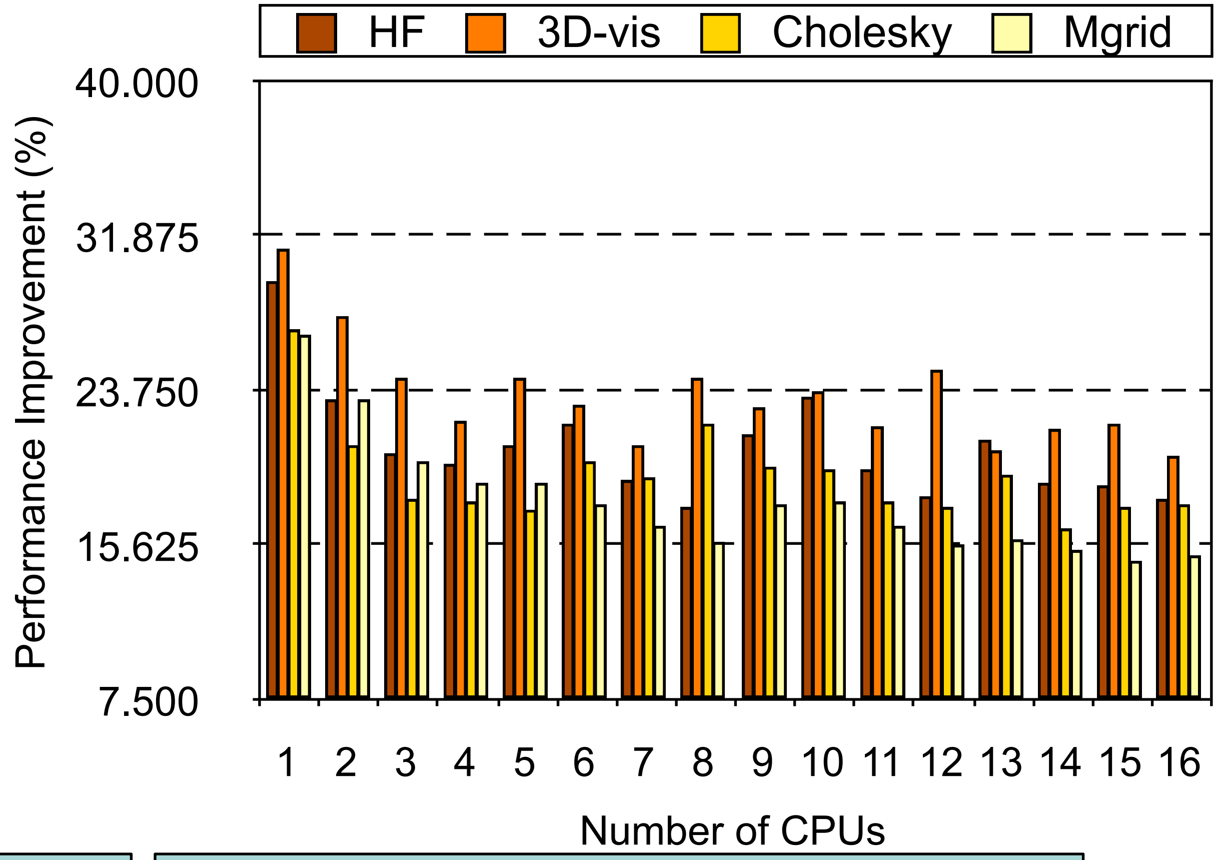


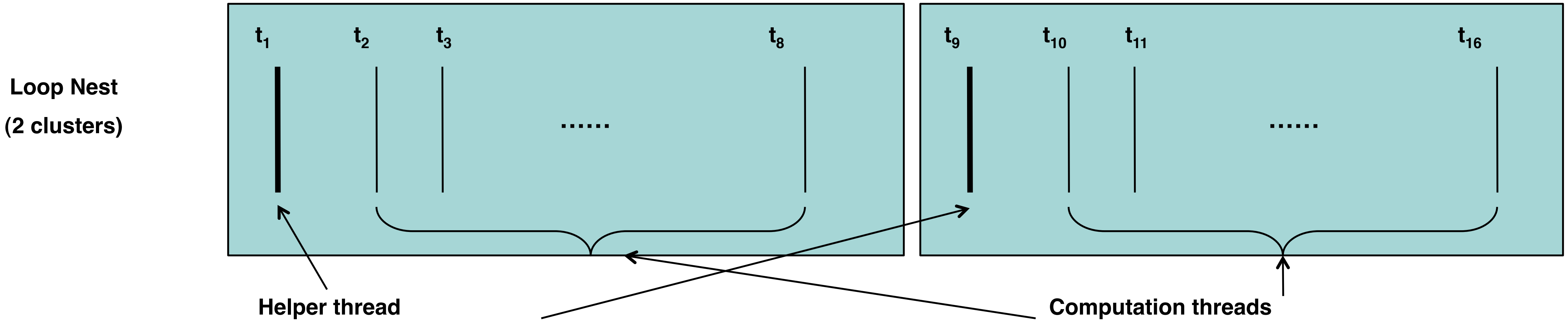


Helper Thread Based I/O Prefetching

- Our approach obtains inter-thread data sharing information using profiling and divides parallel threads into clusters and assigns a separate (customized) I/O prefetcher thread for each cluster







Loop Nest

(2 clusters)

Helper thread

Computation threads

Publications

- Wei-keng Liao and Alok Choudhary. "Dynamically Adapting File Domain Partitioning Methods for Collective I/O Based on Underlying Parallel File System Locking Protocols". To appear in SC08.

- Arifa Nisar, Wei-keng Liao, and Alok Choudhary. "Scaling Parallel I/O Performance through Delegation and Cooperative Caching". To appear in SC08.

- Ozcan Ozturk, Seung Woo Son, Mahmut Kandemir, and Mustafa Karakoy. "Prefetch Throttling and Data Pinning for Improving Performance of Shared Caches". To appear in SC08.

- Seung Woo Son, Sai Prashanth Muralidhara, Ozcan Ozturk, Mahmut Kandemir, Ibrahim Kolcu, and Mustafa Karakoy. "Profiler and Compiler Assisted Adaptive I/O Prefetching for Shared Storage Caches". To appear in PACT08.